IN 32

151404

P.17

NASA Technical Memorandum 106092

Real-Time Transmission of Digital Video Using Variable-Length Coding

Thomas P. Bizon, Mary Jo Shalkhauser, and Wayne A. Whyte, Jr.
Lewis Research Center
Cleveland, Ohio

Prepared for the
Data Compression Conference (DCC '93)
sponsored by the Institute of Electrical and Electronics Engineers
Snowbird, Utah, March 30–April 1, 1993

NASA

Real-Time Transmission of Digital Video Using Variable-Length Coding

Thomas P. Bizon, Mary Jo Shalkhauser, and Wayne A. Whyte, Jr.
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, OH 44135

1.0 Introduction

Huffman coding is a variable-length lossless compression technique where data with a high probability of occurrence is represented with short codewords, while "not-so-likely" data is assigned longer codewords. Compression is achieved when the high-probability levels occur so frequently that their benefit outweighs any penalty paid when a less likely input occurs. One instance where Huffman coding is extremely effective occurs when data is highly predictable and differential coding can be applied (as with a digital video signal). For that reason, it is desirable to apply this compression technique to digital video transmission; however, special care must be taken in order to implement a communication protocol utilizing Huffman coding.

This paper addresses several of the issues relating to the real-time transmission of Huffman-coded digital video over a constant-rate serial channel. Topics discussed include data rate conversion (from variable to a fixed rate), efficient data buffering, channel coding, recovery from communication errors, decoder synchronization, and decoder architectures. A description of the hardware developed to execute Huffman coding and serial transmission is also included. Although this paper focuses on matters relating to Huffman-coded digital video, the techniques discussed can easily be generalized for a variety of applications which require transmission of variable-length data.

2.0 Summary of Compression Algorithm

A compression algorithm was developed at NASA Lewis Research Center to process 8-bit samples of the composite color NTSC[1] video signal taken at four times the color subcarrier frequency. After compression, the amount of digital data required for video transmission is reduced by over 75% with virtually no visible degradation in picture quality. The algorithm is based on differential pulse code modulation (DPCM), but additionally utilizes a non-uniform quantizer, non-adaptive predictor, and multi-level Huffman coder to reduce the data rate substantially below that achievable with conventional DPCM. A block diagram of the algorithm can be found in figure 1.

1) National Television Systems Committee

The differential (DPCM) portion of this algorithm performs all necessary operations to make a prediction of the incoming pixel and produce a difference value. This difference is then grouped into one of thirteen quantization levels (QL) and passed to the Huffman coder. The coder contains thirteen individual Huffman trees (one for each quantization level), with each tree consisting of thirteen variable-length codewords. This multi-level structure facilitates dynamic adaptation of the QL probabilities based on the previous input. For example, when an input of QL 4 is received, the coder will select Huffman tree 4 for the next input. Huffman tree 4 represents the probability of occurrence for all QL values given that the previous input was QL 4. In this way, the multi-level Huffman coder is self-optimizing; therefore, higher compression ratios are achieved than those obtainable with a single-tree coder[2]. Once Huffman coding is completed, the data is fully compressed and ready for serial transmission. A complete description of the algorithm may be found in reference [1] or [2].

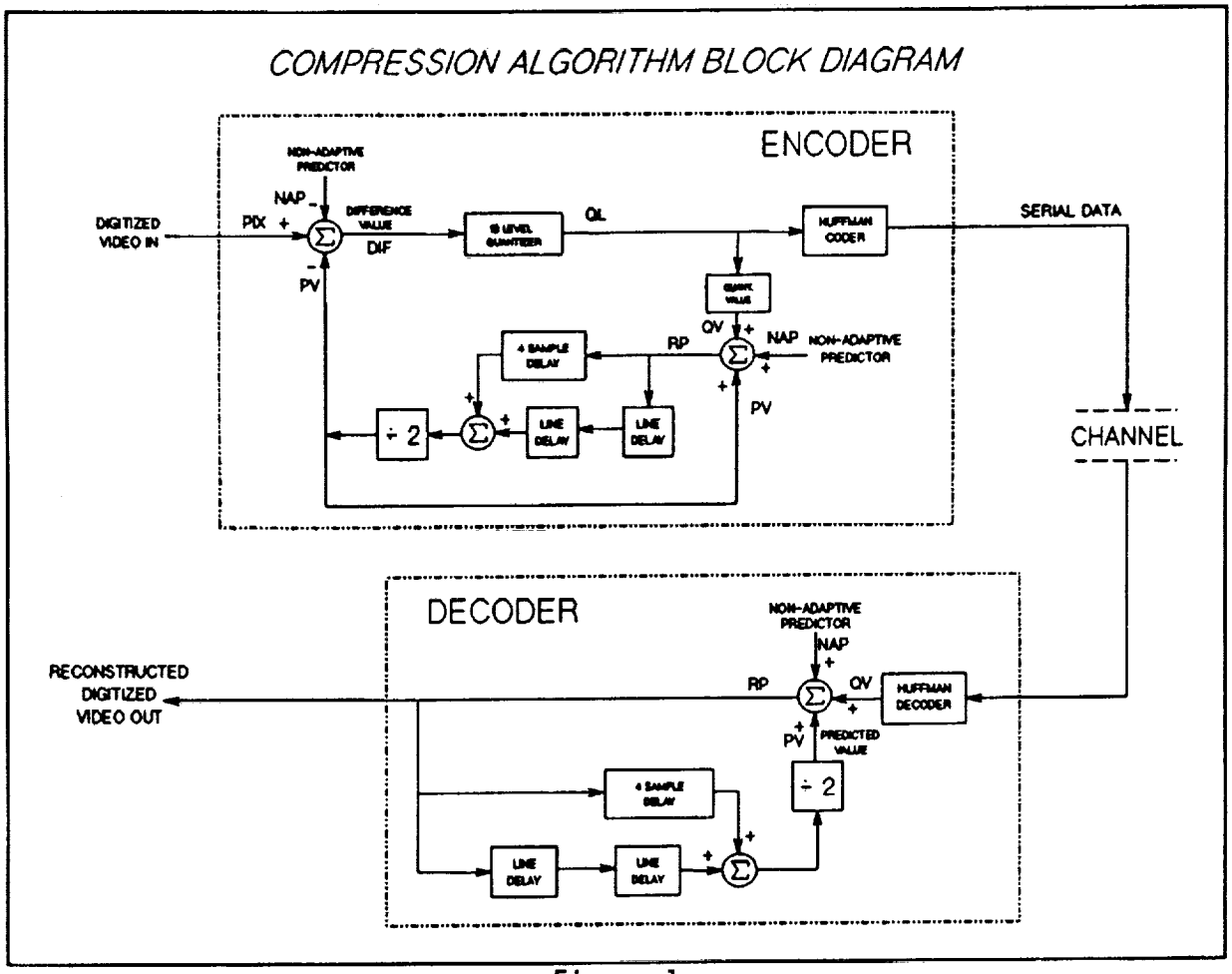

Figure 1

2) Simulation results have demonstrated a one-half bit per pixel improvement in compression performance for the multi-level Huffman coder over a single-tree coder.

3.0 Encoder Implementation Issues

3.1 Data Rate Conversion

Because of the variable-length representation of information, the output data rate of the Huffman coder varies with each quantization level input. Since the data must be transmitted over a constant-rate communications channel, it is necessary to incorporate a buffer to smooth these rate variations and create a constant-rate data stream. This conversion is most easily accomplished using a first-in/first-out memory (FIFO) to absorb the differences between data rates. Depending on video content, however, differences between the rates can accumulate, and there is a danger of overflowing or underflowing the FIFO. For this reason, it is necessary to include additional protection against FIFO overflow and underflow. Since the channel rate is fixed, the mechanism used must adaptively reduce or increase the amount of data entering the FIFO. To prevent overflow, reduction of input data is accomplished using an alternate (coarser) quantization scheme generated by regrouping the original thirteen levels into five. This scheme fosters data reduction using the most probable (shortest) Huffman codes, while still maintaining reasonable picture quality. To prevent underflow, augmentation of input data is done by eliminating the compression effect of the Huffman coder and transmitting the quantization level directly (at a rate of four bits per QL). In this case, picture quality remains the same as for the normal algorithm since no quantization information is changed. To determine the processing mode used (normal, data reduction, or data augmentation), the FIFO fill-level is evaluated at the beginning of each video line, and the mode is selected based on preset thresholds.

An intelligent data rate buffer was developed to convert the variable-rate output of the Huffman coder to a constant-rate input for the serial transmission channel. Details of the implementation can be found in section 5.2.

3.2 Efficient Data Buffering

As described above, it is necessary to incorporate a data rate buffer to convert the variable-rate output of the Huffman coder to a constant-rate input for the serial transmission channel. Due to the variable-length representation of the data, however, it is impossible to efficiently use the buffer memory without some additional pre-storage processing. One simple operation to facilitate efficient storage is to pack (group) the variable-length Huffman codes into words of constant-length which can be stored utilizing the full width of the memory. Efficient data buffering is achieved because only valid data (and no overhead) is stored in the memory and all available memory width is used. A method was developed to perform the data packing, and a sample of the packer algorithm (using a word length of 16-bits) is shown in figure 2.

Referring to the figure, input to the data packer consists of a "Huffman Code" and the "Code Length". Upon receipt of this input, the packer will transfer the code into a constant-length word beginning at the first empty location. This location is easily derived

using the internal "Bits Filled" information. Once a word is completed (16 or more bits are filled), that word is transferred to the data rate buffer, and construction of the subsequent word begins. It is possible for a Huffman code to be split between two of the constant-length words; however, system performance is not affected since the data will be transmitted serially. The "Output Grouping" of the sample input sequence is shown in the figure.

A data packing circuit was developed as part of the intelligent data rate buffer described above. Details of the implementation can be found in section 5.2.

Sample Sequence for Data Packer

Huffman Code	Code Length	Bits Filled	Word Done
0001	4	0	1
01	2	4	0
1	1	6	0
001	3	7	0
000001	6	10	0
00000001	8	0	1
1001	4	8	0
00000000001	11	12	0
00001	5	7	1
0000001	7	12	0
XXXXXXXXXX	XX	3	1

Output Grouping

4	2	1	3	6
8			4	4 of 11
7 of 11			5	4 of 7
3 of 7	XX			

Note: Numbers represent the length of the Huffman code in bits.

Figure 2

4.0 Decoder Implementation Issues

4.1 Decoder Synchronization

The video decoder receives the serial data stream from the transmission channel and must reverse the compression procedure performed by the video encoder to reconstruct the video image. This requires that the decoder adapt decompression protocols to account for changes in processing mode (normal, data reduction, or data augmentation) and DPCM predictive techniques (see reference [1]). These changes occur exclusively at video line boundaries; hence, the decoder needs only to derive this information from the serial data stream for proper decompression.

Line boundaries are difficult to recognize, however, because the variable-length nature of the Huffman codes results in a variable number of bits per video line. The line boundaries can be distinguished after a complete line is processed by the Huffman decoder; however, the decoding process is extremely vulnerable to bit-errors induced by the transmission channel. When a bit-error occurs in a Huffman code, proper decoding is impossible and the decoder can lose synchronization with correct codeword boundaries and, therefore, with correct line boundaries.

To maintain synchronization at the video decoder, the video encoder inserts a unique word at the beginning of each line and frame. The unique words were chosen to maintain a reasonable level of overhead and hardware complexity, while minimizing the possibility of unique word occurrences in the video data. Simulations were run using a variety of still images to empirically choose the "best" unique words. The unique words are immediately followed by two mode bits which allow the video decoder to properly decode the line using the normal, data reduction, or data augmentation mode. Additionally, the unique words can be utilized by the video decoder for error detection and concealment as described in the following section.

4.2 Channel Coding and Recovery from Communication Errors

One drawback to variable-length coding, such as Huffman coding, and to predictive compression techniques, such as DPCM, is non-graceful degradation when bit-errors occur on the transmission channel [3]. As described previously, bit-errors in Huffman-coded data result in incorrectly decoded pixel values and possible loss of codeword synchronization. These incorrect pixel values can then propagate through the video image because future prediction values will be based upon erroneous prior values. The result is poor reconstructed image quality.

Three different techniques can be applied to minimize the impact of channel errors on the image quality. Forward error correction can be used to protect the transmitted data from channel errors, windowing can be used to minimize false unique word detects, and line substitution can be used for error concealment to reduce the propagation of errored pixels through the video image.

Forward error correction (FEC) can be used to protect any data set, including Huffman-coded image data, from errors induced by a noisy transmission channel. An FEC encoder adds redundant bits to the serial image data, and an FEC decoder uses those bits to detect and correct bit-errors that occur within the data stream. For our application, a block FEC codec was chosen over a convolutional codec because of the error correction performance for minimal data overhead. In particular, a rate 239/255 Reed-Solomon codec was used. Simulations show that this codec will reduce a 5×10^{-6} bit-error-rate (BER) channel to a 7.5×10^{-34} BER channel (see table 1).

Channel Error Rate	
Without Coding	With Coding
1×10^{-2}	4.4×10^{-5}
1×10^{-3}	3.1×10^{-13}
1×10^{-4}	3.7×10^{-22}
1×10^{-5}	3.8×10^{-31}
5×10^{-6}	7.5×10^{-34}
1×10^{-6}	$< 10^{-35}$

Table 1: FEC Simulation Results

While the use of this FEC codec can greatly reduce the probability of channel bit-errors, it also has a few disadvantages. FEC coding adds redundancy to the compressed image data, effectively nullifying some of the compression algorithm gains. The increased redundancy also means that the channel data rate is increased. Another disadvantage is the increased hardware complexity introduced by the FEC coding and decoding circuits.

When FEC coding is not desirable due to limited bandwidth availability or when the FEC decoder fails to correct all channel errors, two additional techniques (described below) can be used by the video decoder to limit the impact of the errors.

As discussed previously, unique words are utilized to maintain synchronization at both the video line and frame boundaries. Although the unique words were chosen to minimize their possibility of occurrence in the video data, simulation results have shown that even the best unique words can sometimes occur randomly within the serial data stream. In addition, bit-errors induced on the transmission channel can cause additional unexpected occurrences of unique words throughout the video data. In order to reduce the incorrect detection of these false unique words, windowing techniques can be applied that allow unique word detects to occur only near their expected location in the data (i.e. after most of a video line has been decoded).

In addition to decoder synchronization, the unique words can be used for error detection and concealment. Due to the chosen sampling rate, there are 910 pixels per video line[3]. When errors occur in the Huffman-coded data, the number of decoded pixels per line can vary; therefore, the unique word location, rather than the pixel count, is the determining factor for line boundary location. Following a unique word detect, the Huffman decoder begins decoding the serial data. If the next unique word occurs before or after 910 pixels have been decoded, then an error has occurred somewhere on the line.

Because the video receiver must have exactly 910 pixels per line, the line lengths have to be adjusted when an incorrectly decoded Huffman code results in a line that is longer or shorter than 910 pixels. In addition, the line contains pixel errors. When the incorrect line is used by the predictive circuits to reconstruct the video image, the errors will propagate throughout the rest of the video frame and the resultant image quality will be poor. An error concealment technique is used to adjust the line lengths to 910 pixels per line and also to mask errors in the line.

When an errored line is detected by the unique word location, error concealment is performed by substituting the most recent error-free line having the same subcarrier phase characteristics as the current incorrect line. Although the substituted line is not likely to be an exact match for the original line, it is likely to be very similar. The result is that some (relatively small) error is introduced in the video reconstruction with the substitution; however, the large adverse effects of decoding errors are masked.

5.0 Implementation of Encoder Hardware

5.1 Huffman Coder

The Huffman coder performs all operations necessary to generate the variable-length codes using the multi-level tree structure previously discussed. Since the probability tree used at any given time is based only on the previous quantization level (QL), it is possible to implement the complete multi-level coder using a single look-up table addressed by both the current and previous QLs. The output of this look-up table is the corresponding Huffman code and a length indicator denoting the number of bits used by that code. This information is then transferred to the intelligent data rate buffer for efficient storage.

In addition to variable-length coding, the Huffman coder look-up table is used to implement the data augmentation procedure and the unique word/mode bits insertion. Both operations are accomplished by addressing "special" sections of the look-up table at the appropriate times. The data augmentation section of the table contains "Huffman codes" that are equal to the input quantization level---effectively eliminating the coding process. The unique word/mode bits section of the table is subdivided into separate areas

3) 4 x 3.579545 Msamples/sec x 63.56 usec/line = 910 samples/line

containing line or frame unique words and different combinations of mode bits. A block diagram of the complete Huffman coder circuit can be found in figure 3.

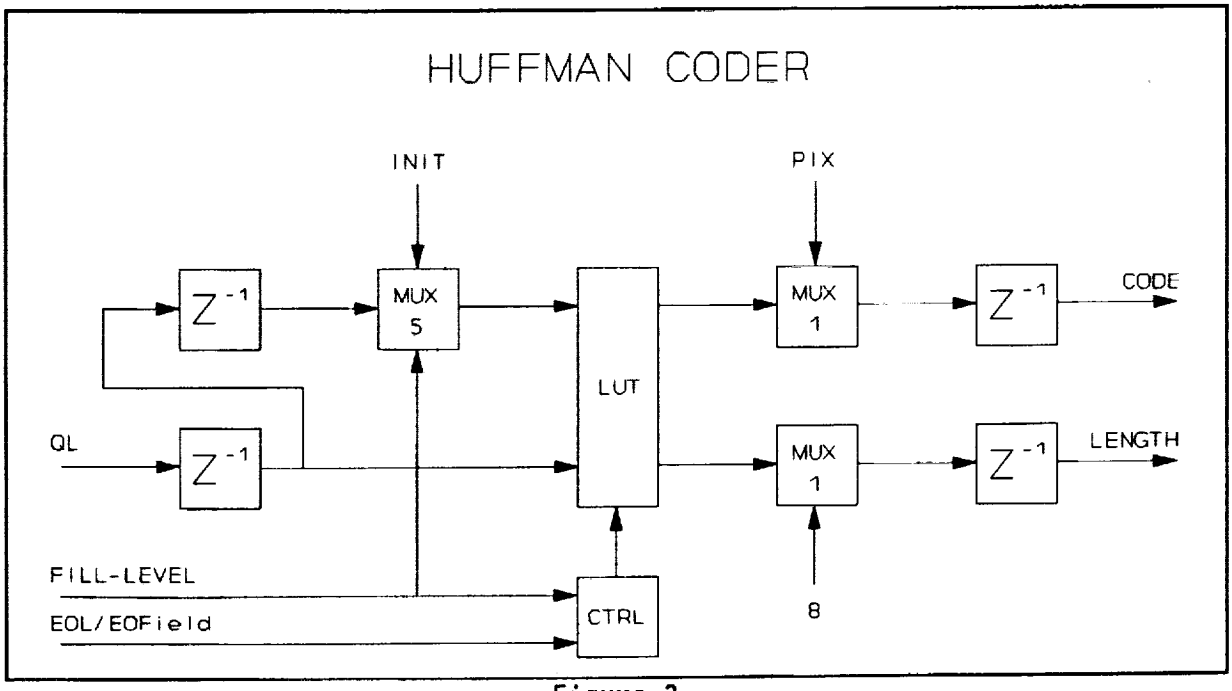

Figure 3

Referring to the figure, input to and output from the look-up table is governed by the Huffman coder controller and various multiplexers. The controller regulates access to the appropriate section of the look-up table for unique word/mode bit insertion using control signals from both the DPCM hardware ("EOL/EOField") and the data rate buffer ("fill-level"). The controller was implemented as a state machine using a programmable logic device (PLD). The multiplexers used implement the initialization and data augmentation procedures of the compression algorithm. MUX1 allows transmission of the incoming pixel uncompressed (necessary for initialization of the algorithm), while MUX5 controls addressing of the look-up table (for initialization and data augmentation). Because the functions of these multiplexers are very similar to those of DPCM quantizer multiplexers, most control signals can easily be derived from the quantizer controller. Additional MUX5 control to execute the data augmentation protocol is supplied by the "fill-level" signal from the data rate buffer.

5.2 Intelligent Data Rate Buffer and Data Packer

The intelligent data rate buffer is used to compensate for differences between the variable-rate Huffman coder output and the constant-rate serial channel. As discussed previously, data rate buffering is accomplished using a FIFO memory with additional safeguard circuitry to prevent underflow or overflow of the FIFO. A block diagram of the data rate buffer is shown in figure 4.

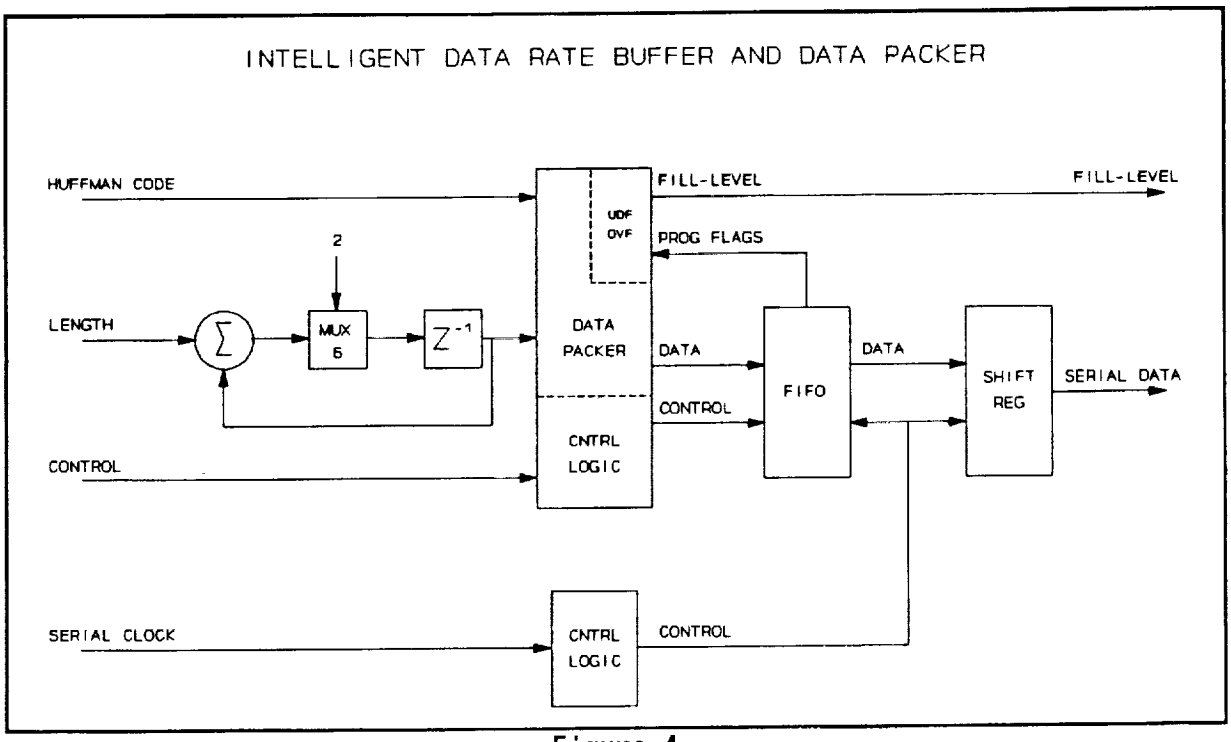

Figure 4

The central unit of the data rate buffer is, of course, the FIFO. For this application, the FIFO chosen has an 18-bit data bus (only 16 bits are used), storage capacity of 8192 words, and both a programmable almost-empty flag and a programmable almost-full flag. Small variations between the data rates of the Huffman encoder and the serial channel will be absorbed with the FIFO; however, accumulated variations will require the programmable flags to act as the protection (discussed previously) to prevent FIFO underflow and overflow. These flags are each programmed to mark a fixed fill-level of the FIFO and then polled at the beginning of each video line to select the processing mode for that line as shown in table 2. The levels chosen for the almost-empty and almost-full flags are 127 words (2032 bits) from empty and 2045 words (32720 bits) from full. These levels were experimentally determined to prevent both FIFO underflow and overflow (with significant margin) over a wide range of test images.

In order to perform the above initialization and operation procedures of the FIFO, it is necessary to implement a FIFO input controller. Specific functions of this controller consist of presetting the programmable flags, initiating data storage, polling the fill-level of the FIFO, and selecting the processing mode. This controller was realized as a state machine and implemented with programmable logic (part of the data packer discussed below).

As described previously, it is important to pack (group) the variable-length data into constant-length words to efficiently utilize the FIFO memory. This is accomplished using the data packer, which receives the Huffman codes and generated packing information and groups the codes into 16-bit words. The packing information consists of the bits

filled in a partial 16-bit word and a flag to indicate when the word is completed (see example in section 3.0). It is generated using a four-bit adder to sum the length indicators from the Huffman coder, with the carry output marking when a full word has been constructed. As done in the Huffman coder, a multiplexer (governed by the FIFO input controller) is used for initialization of the packing information.

Almost Empty	Almost Full	Processing Mode
0	0	Normal
1	0	Data Augmentation
0	1	Data Reduction

0 = Signal not asserted, 1 = Signal asserted

Table 2: Operation of FIFO Underflow/Overflow Safeguard

The packer, itself, is equation-intensive; hence, it was necessary to implement this unit in a Mega-PLD. The PLD chosen contains 84 pins, 128 flip-flops, and over 500 product terms. Since this device is so large, the state machine previously discussed was also implemented within the PLD.

Interfacing between the FIFO and serial channel is accomplished via a shift register, which will convert 16-bit parallel data into a serial bit stream. Like before, control circuitry was developed to supervise required operations, specifically to initiate FIFO recall and to derive various control signals (for both the shift register and the FIFO) from the serial clock. As with other control circuitry, the FIFO output controller was implemented as a state machine in a PLD with some support logic.

6.0 Decoder Architectures

The Huffman decoder circuit receives serial data from the transmission channel or the FEC decoder, detects line and frame unique words, reverses the Huffman coder operation, and performs error detection and concealment. Two different approaches to the Huffman decoder circuit, parallel decode and serial decode, have been investigated. Each of the approaches has advantages and disadvantages; hence, the best approach is dependent on the application. Both of the above decoder architectures are discussed in detail in the following sections.

6.1 Parallel Huffman Decoder

In the parallel Huffman decoder approach, the serial data stream is converted to parallel data and presented to a look-up table for Huffman decoding. The proposed parallel Huffman decoder architecture (shown in figure 5) uses concepts similar to those used in the Huffman coder/data packer circuits discussed in section 5.0.

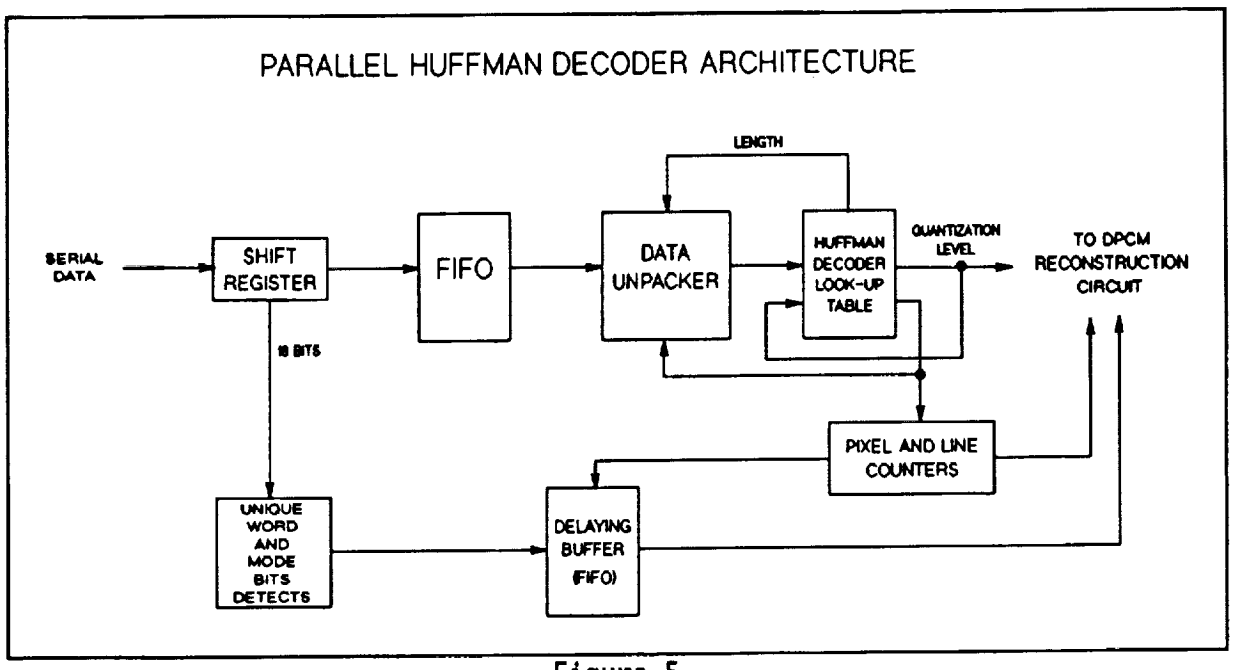

Figure 5

Huffman-coded serial data is received from the transmission channel or from the FEC decoder. First, the unique words are detected for initialization purposes and for mode selection. Then, the unique words and the mode bits are removed from the data stream. The remaining data is stored in the rate buffer FIFO. Additional circuits are required to maintain the line and frame boundaries and the mode information so that the video decoder can remain aligned with the compressed data in the FIFO.

Data is read from the FIFO and loaded into the data unpacker, which performs the function of a barrel shifter. At the uncompressed pixel rate, the data unpacker presents an 11-bit word to the Huffman decoder look-up table (LUT). Since the maximum Huffman code length is 11 bits in this application, the 11-bit word is guaranteed to contain at least one Huffman code.

The 11-bit word is only part of the address bits in the Huffman decoder LUT. Additional address bits are required because the multi-level Huffman codes are dependent on the previous quantization level (QL); therefore, the decoded QL must be fed back to address the look-up table. With the 11-bit word and the previous QL value as address bits, the

LUT outputs the correct QL value for the Huffman code and a code length value. This code length is fed back to the data unpacker which shifts the data by that length and outputs a new 11-bit word. To maintain at least 11 bits of valid data in the shift register at all times, the data unpacker receives additional data from the FIFO as needed.

The major advantage of this parallel Huffman decoder approach is that the FIFO is small and equal in size to the encoder FIFO because it stores the compressed data. Some penalty is paid, however, in the circuit complexity of the data unpacker and the memory size of the LUT. Another advantage is that the Huffman decoder LUT can operate at the relatively slow pixel clock rate. A significant disadvantage of this approach appears when bit-errors are possible in the system. As discussed previously, bit-errors in the serial data stream will result in incorrectly decoded Huffman codes, subsequently resulting in loss of codeword and line synchronization. If the line lengths are incorrect, the error concealment techniques are virtually impossible to implement. In addition, when errors occur, the actual time to decode a video line will vary. This will result in significant problems with the video decompression circuits following the Huffman decoder.

The parallel Huffman decoder approach is straight-forward and memory-efficient, and is recommended for applications in which channel transmission errors are not a factor.

6.2 Serial Huffman Decoder

As in the parallel decoder approach, the serial decoder (shown in figure 6) utilizes a LUT for Huffman decoding. In this case, the LUT stores each of the Huffman code trees and performs a tree search to decode each Huffman code. A pointer is initially set to the top node of the tree. As each serial bit is received, the pointer branches down to the next node of the tree. The direction of each branch depends on the value of the serial bit. The Huffman code tree is traversed in this manner until a valid Huffman code has been detected. When the end of a valid Huffman code is reached, the LUT outputs an end-of-code flag and a quantization level. The next address is set to zero to move the pointer to the top of the next tree. The quantization level (QL) is fed back as an address for the LUT to select the proper Huffman code tree for decoding. To handle different processing modes (normal, data reduction, or data augmentation), the mode bits are used as the most significant address bits to the LUT. The Huffman decoder operation may be more easily understood by examining the example in figure 7.

The serial Huffman decoder LUT is preceded by an 18-bit shift register. The serial output of the shift register is fed directly to the Huffman decoder LUT, and the parallel output is fed to line and frame unique word detect circuits and the mode decode circuit. Each time a unique word is detected, the mode bits are tapped off to be used by the rest of the decoder circuitry, and the Huffman decoder is disabled until all 18 bits are purged from the shift register so that the unique words and mode information are not incorrectly interpreted as Huffman data.

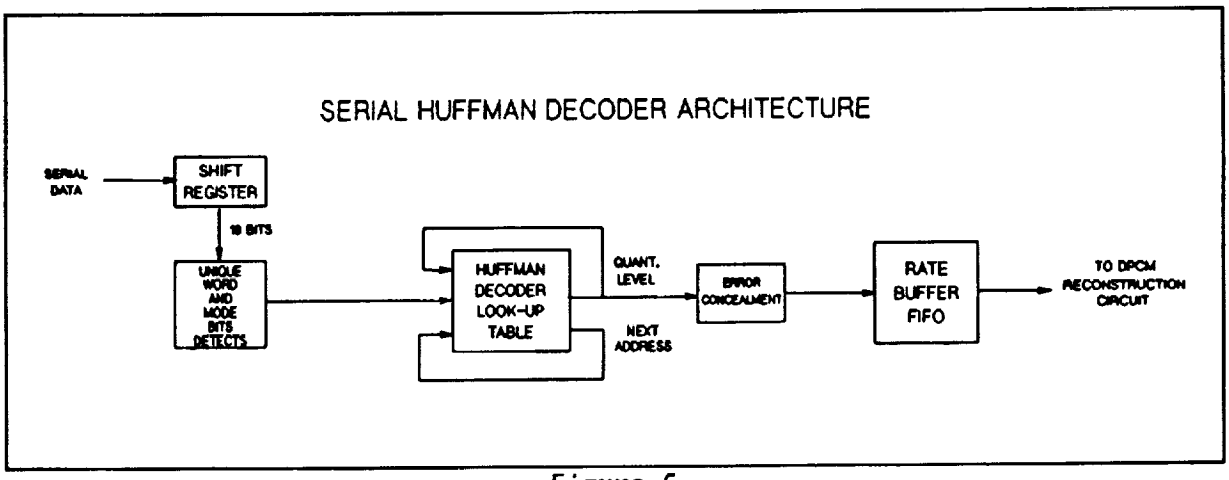

Figure 6

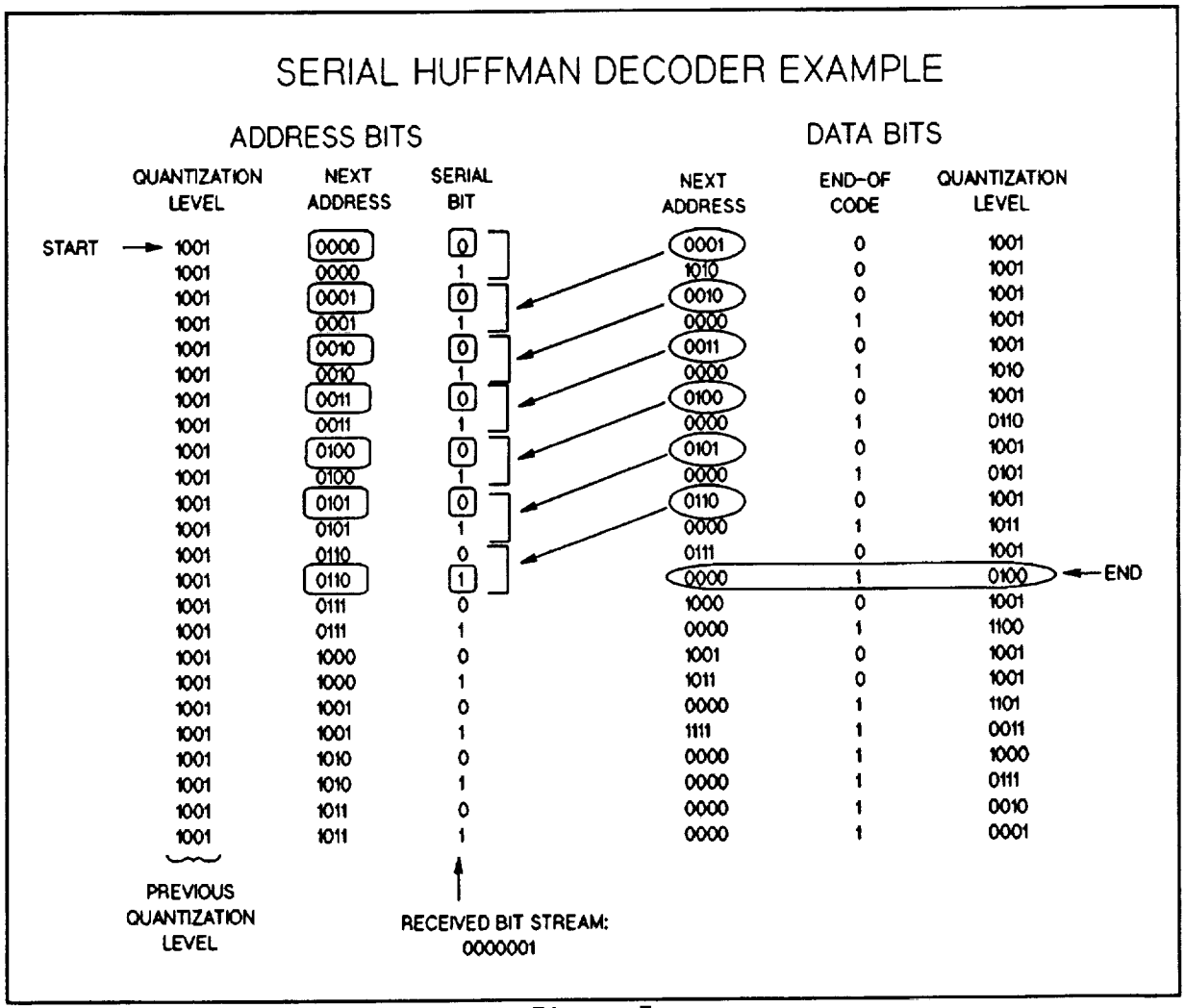

Figure 7

The serial Huffman decoder architecture uses the location of the unique words and a pixel counter to detect if a line contains an error. As each codeword is decoded, a pixel counter is incremented to count the number of pixels per line. If a unique word window is used, a window is opened toward the end of the current video line to begin looking for the start of the next video line. If no errors occurred on the line, there will be exactly 910 pixels on that line. If there are errors, however, the Huffman decoder will most likely lose synchronization with the Huffman codeword boundaries resulting in line lengths greater than or less than 910 pixels.

To compensate for Huffman decoder errors, an error concealment circuit (ECC) follows the Huffman decoder. When an errored line is detected, the ECC will replace that line with the most recent line of similar content in order to adjust line length and mask error effects. For this application, neighboring lines are similar in content, with each line having the same phase relationship as the line two previous to it. The ECC (see figure 8) consists of three interconnected FIFOs, each 910 words deep.

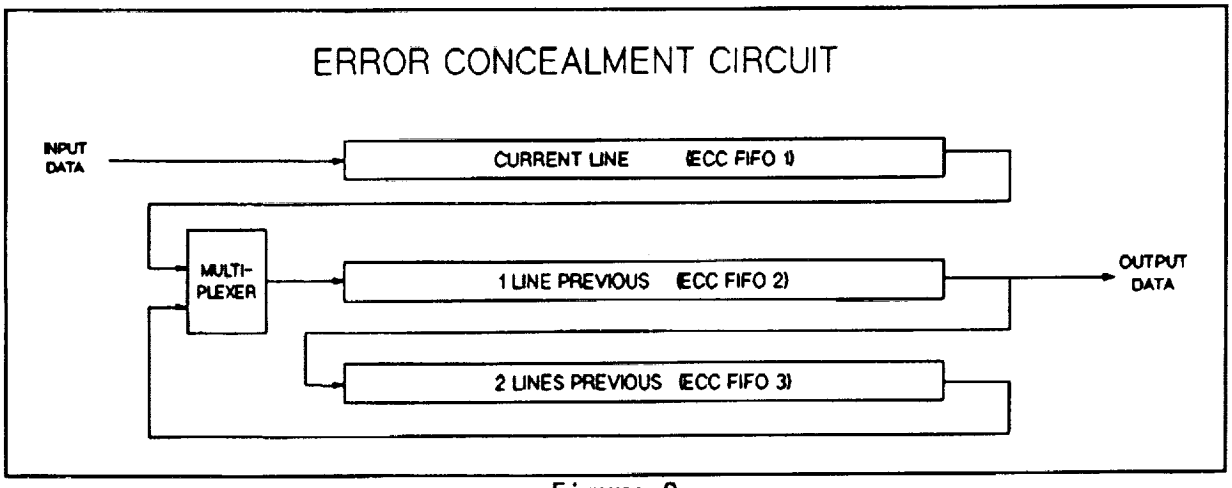

Figure 8

The Huffman decoder output is loaded directly into the first ECC FIFO. If the current line is error-free, the second ECC FIFO is loaded with the contents of the first ECC FIFO. Otherwise, the second ECC FIFO is loaded with the contents of the error-free third ECC FIFO. The output ECC FIFO 2 is sent to the remainder of the video decoder circuitry which is responsible for rate buffering and reconstructing the video signal using predictive techniques.

The major advantage of the serial Huffman decoder approach is its ability to detect and conceal errors that occur within the Huffman-coded data. The rate buffer FIFO is larger, however, because it does not store compressed data as with the parallel decoder approach. Another advantage is that the decoding time for an errored line and a non-errored line is the same. A disadvantage is that the LUT must operate at the relatively high serial bit rate. The error detection and concealment circuits also increase the hardware complexity.

The serial decoder architecture is recommended for applications in which channel errors are likely to occur. The compression algorithm (described in section 2.0) will be used for satellite communication applications; therefore, the serial Huffman decoder architecture has been chosen for implementation.

7.0 Conclusion

A digital video compression algorithm utilizing Huffman coding has been developed at NASA Lewis Research Center to facilitate transmission of video data. In order to implement a communications system utilizing coding of this type, however, it is necessary to consider several issues such as data rate conversion (from variable to a fixed rate), efficient data buffering, channel coding, recovery from communication errors, and decoder synchronization. Each of these topics was addressed in detail in this report. Additionally, a description of hardware developed to execute Huffman coding and serial transmission was included in the report.

Development of decoder hardware is continuing. Two different Huffman decoder approaches, parallel decode and serial decode, have been described and compared. The best approach is dependent on the particular application. The parallel Huffman decoder is memory-efficient and easier to implement than the serial approach, but it is not suitable for applications with possible transmission channel errors. The serial Huffman decoder approach is more complex, but can be used in applications with noisy channels because transmission errors can be detected and masked. For evaluation of algorithm performance, the compressed data will be transmitted over an experimental satellite link; therefore, the serial Huffman decoder approach was chosen for implementation.

References

1. Bizon, T. P.; Whyte, W. A.; Marcopoli, V. R.: Real-Time Demonstration Hardware for Enhanced DPCM Video Compression Algorithm. NASA Technical Memorandum 105616, March 1992.

2. Shalkhauser, M. J.; Whyte, W. A.: Digital CODEC for Real-Time Processing of Broadcast Quality Video Signals at 1.8 Bits/Pixel. NASA Technical Memorandum 102325, November 1989.

3. Adkins, K.; Shalkhauser, M. J.; Bibyk, S.: Digital Compression Algorithms for HDTV Transmission. 1990 IEEE International Symposium on Circuits and Systems (ISCAS), May 1990.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 1993	Technical Memorandum

4. TITLE AND SUBTITLE
Real-Time Transmission of Digital Video Using Variable-Length Coding

5. FUNDING NUMBERS
WU-144-10-10

6. AUTHOR(S)
Thomas P. Bizon, Mary Jo Shalkhauser, and Wayne A. Whyte, Jr.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135-3191

8. PERFORMING ORGANIZATION REPORT NUMBER
E-7730

9. SPONSORING/MONITORING AGENCY NAMES(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, D.C. 20546-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA TM-106092

11. SUPPLEMENTARY NOTES
Prepared for the Data Compression Conference (DCC '93) sponsored by the Institute of Electrical and Electronics Engineers, Snowbird, Utah, March 30-April 1, 1993. Responsible person, Thomas P. Bizon, (216) 433-8121.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified - Unlimited
Subject Category 32

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)
Huffman coding is a variable-length lossless compression technique where data with a high probability of occurrence is represented with short codewords, while "not-so-likely" data is assigned longer codewords. Compression is achieved when the high-probability levels occur so frequently that their benefit outweighs any penalty paid when a less likely input occurs. One instance where Huffman coding is extremely effective occurs when data is highly predictable and differential coding can be applied (as with a digital video signal). For that reason, it is desirable to apply this compression technique to digital video transmission; however, special care must be taken in order to implement a communication protocol utilizing Huffman coding. This paper addresses several of the issues relating to the real-time transmission of Huffman-coded digital video over a constant-rate serial channel. Topics discussed include data rate conversion (from variable to a fixed rate), efficient data buffering, channel coding, recovery from communication errors, decoder synchronization, and decoder architectures. A description of the hardware developed to execute Huffman coding and serial transmission is also included. Although this paper focuses on matters relating to Huffman-coded digital video, the techniques discussed can easily be generalized for a variety of applications which require transmission of variable-length data.

14. SUBJECT TERMS
Data compression; Variable-length coding; Huffman coding; Digital video

15. NUMBER OF PAGES
A03

16. PRICE CODE
16

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	